PERCOM

PERCOM 6809 SYSTEM MONITOR

PSYMON(TM)

USERS MANUAL

PERCOM DATA COMPANY
211 N. KIRBY
GARLAND, TEXAS 75042

PSYMON
PERCOM SYSTEM MONITOR FOR THE 6809

Percom Data Company, Inc.
211 N. Kirby
Garland, Texas 75042
(214) 272-3421

INTRODUCTION

PSYMON, the Percom SYstem MONitor for the 6809, is a simple 1K operating system designed for the Motorola 6809 microprocessor. While it provides commands for program loading and saving, memory and register examine/change, and breakpoint management, the true power of PSYMON is in its structure and extensibility.

PSYMON was designed to be as easy as possible to interface to regardless of the hardware environment. It may be highly customized and extended due to its unique "look-ahead" and device independent I/O structure. This adaptability was the result of the use of structured techniques in the design and programming of PSYMON. The members of the design team were Harold Mauch, Mike Foreman, Byron Seastrunk, Cliff Rushing, and Jim Stutsman. All of these team members have extensive experience with a variety of monitors for the MC6800 from which to draw.

DESCRIPTION OF COMMANDS

When PSYMON first receives control (usually through the power-on vector of the 6809 processor) it initializes its RAM areas, configures its console, and looks ahead for a second PROM (more about this later). At this time PSYMON will prompt with 'CMD?' and wait for the input of a legal command. All commands consist of a single letter. Some require parameters in the form of address or data. Whenever hexadecimal data is input to PSYMON, it is accepted according to a simple scheme. First, any non-hex character (other than 0-9 or A-F) terminates the hex entry. Certain "terminator" characters may have special meaning depending on the command. Second, leading zeroes are assumed on all entries shorter than the required size. For example, entry of FE as a parameter for an address would be interpreted as 00FE. Finally, if more digits are entered than are expected, only the last ones entered are used. For example, if 12345 is entered when a single byte is expected, the value used will be 45.

Command Set Summary

M <ADDRESS>	- MEMORY EXAMINE/CHANGE
G <ADDRESS>	- GO TO ADDRESS
R <REGISTER>	- REGISTER EXAMINE/CHANGE
L	- LOAD PROGRAM (FROM TAPE)
S <START> <END>	- SAVE PROGRAM (TO TAPE)
B <ADDRESS>	- SET/LIST BREAKPOINTS
U <ADDRESS>	- UNSET BREAKPOINTS
Z	- JUMP TO ADDRESS C000 (HEX)

M <address> - Memory examine and change

The command waits for an address to be entered. If a valid hex address is NOT entered, the LAST address examined is used (initially 0). This feature minimizes user frustration when inadvertently terminating a Memory Examine/Change sequence. It is also useful if you wish to repeatedly examine the same address (such as an I/O port).

First the address is displayed, followed by its contents in hex. The contents may be changed by entering a new value followed by a terminating character. If a new value is entered it is written into memory and verified. If the data did not store as expected, a '?' is displayed. Whether or not data was changed, the terminating character of the user entry is then examined. If the terminating character is '^', the address and content of the memory byte PRECEDING the one just examined will be displayed. The command then executes as previously described. If the terminating character is a CARRIAGE RETURN, the Memory Examine/Change is ended and control returns to the command prompt. Any OTHER terminating character will cause the address and content of the memory byte FOLLOWING the one just examined to be displayed and the examine/change process continues as described.

Examples:

```
M <TERM>              Displays last memory byte examined
                      (initially 0000)
M 1234<TERM>          Displays memory byte $1234
1234 F8 <SPACE>       SPACE causes display of NEXT byte
1235 F9 3F<SPACE>     F9 changed to 3F, display NEXT byte
1236 FA ^             No change, display PRECEDING byte
1235 3F <CR>          Carriage Return ends Examine/Change
CMD?
```

R <register> - Register examine and change

The command waits for the entry of a register name from the following list:

A - Accumulator A
B - Accumulator B
C - Condition code register
D - Direct page register
X - Index register X
Y - Index register Y
U - User stack pointer
P - Program counter

If no valid register name is entered, all registers are dumped and the command terminates. For a valid entry the contents of the register is displayed and the command waits for a replacement value to be entered. If a new value is entered it replaces the old value. In either case the command terminates and returns to the command prompt.

G <address> - Go to address

If a valid address is entered, it is placed in the Program Counter position on PSYMON's stack. If NO valid address is entered, the value already in the Program Counter position on the stack is used. All of the 6809 registers are loaded from PSYMON's stack (with an RTI instruction) and execution begins at the location pointed to by the program counter. Warning - the first thing user programs must do on receiving control is to establish a system stack (an LDS instruction). The stack space allocated for PSYMON is too limited for many applications. Failure to establish a new stack will result in the destruction of initial register settings.

L - Load a program from cassette

This command starts the cassette by raising the ACIA RTS (Reader Control) line. The tape is then scanned for records in the Motorola S1-S9 format. The load may be terminated in three ways:

1. Reception of an S9 record.
2. Detection of an invalid checksum.
3. Reception of a non-hex character in an S1 record.

In the case of 2 and 3 a '?' will be printed on the console. Note that tape I/O may be tailored to use other devices and techniques. This will be discussed later.

S <start> <end> - Save a program on cassette

The save command waits for user input of the starting and ending addresses of the memory to be saved on cassette. If only one address is entered, only the data at that address is saved. If NO address is entered, no data is saved and the actual save portion of the command is bypassed. Memory data is output to cassette in the standard Motorola S1 format. After all data has been saved the command terminating character entered by the user from the console is analyzed. If the terminating character is a CARRIAGE RETURN an S9 record is output to cassette. Any other terminator will suppress the S9 record. Finally control returns to the command prompt.

Examples:

S 100 3FF	Save memory from address $0100 through $03FF (no CR so no S9 record)
S 1000	Save byte from address $1000
S 500 7FF<CR>	The CR creates an S9 record after the data is saved
S <CR>	Output S9 record (no data)

B <address> - Set/list breakpoints

The command waits for entry of an address. If one is entered, and there is space in the breakpoint table (10 breakpoints maximum), the breakpoint is set and entered in the breakpoint table. In all cases all currently active breakpoints are listed. Warning - DO NOT breakpoint a location which already has a breakpoint. This condition will not be detected and will probably result in error.

U <address> - Unset a breakpoint

This command waits for input of a breakpoint address. If an address is entered the breakpoint table is searched for a match. When found, the breakpoint is removed. If the breakpoint cannot be found no action is taken. If no address is entered ALL active breakpoints are removed. Note - if a breakpoint is encountered during program execution, the breakpoint is automatically removed.

Z - Call PROM routine

This command, a relic from 6800 systems, is provided for user convenience. When entered, it performs a JSR to memory location $C000. Since PSYMON is designed to seek the highest level of existent operating system, this command will only be useful is the simplest systems.

PSYMON OPTIONS

PSYMON offers a rich variety of options which allow it to be tailored for nearly any configuration. This is done using the unique "look-ahead" feature. At power-up or reset, after initializing RAM and configuring the system console I/O device, PSYMON checks memory location F800. If a 7E (JMP instruction) is found PSYMON does a JSR to F800. This allows a user-written routine to alter any or all of the pointers used by PSYMON. To continue using revised RAM information the user routine need only do RTS (return from subroutine). Optionally the user routine may retain control and use PSYMON only for its subroutines.

All I/O in PSYMON uses a data structure known as a DEVICE CONTROL BLOCK (DCB). The DCB allows PSYMON to be relatively I/O device independent by leaving as much of the detail of the actual I/O as possible to the specific I/O device driver. The DCB is simply a table of parameters located somewhere in memory which among other things contains the address of the device driver routine. The Input/Output characteristics of the system may be subtly or radically altered by changing the contents of the DCB or by directing I/O through a different DCB. For example, data normally transmitted to the console terminal may be easily redirected to the printer or a disk. Likewise, a program may be loaded from a modem or disk instead of cassette tape by modifying the tape input DCB or by redirecting the input through another DCB.

The DCB is organized as follows:

Field	Offset	Usage
DCBLNK	0	Forward link in DCB chain (0 if last)
DCBDID	2	ASCII code for device identification
DCBDVR	4	Device driver address
DCBIOA	6	Device I/O address (meaningful to driver)
DCBERR	8	Error status code
DCBEXT	9	Number of extension bytes in DCB
DCBAPP	10	Optional appendage depending on driver

PSYMON itself has a single DCB which is used for all console functions. This DCB is initialized for I/O through an ACIA interface but may be altered since both the DCB and the pointers to the DCB are maintained in RAM. All keyboard input to PSYMON uses the DCB whose address is in CIDCB. Thus by changing this address, the input device alone may be changed. Echo of input characters is through the DCB pointed to by CEDCB. The input character echo is suppressed by setting CEDCB to zero. Output to the console device is through the DCB addressed by CODCB. All tape I/O uses the DCB pointed to by TPDCB. These pointers all initially point to CONDCB, PSYMON's console DCB. Any or all of the pointers may be changed by a user routine.

All of the hardware interrupts are vectored through addresses in PSYMON's RAM. SWI3V, SWI2V, and SWIV handle the various types of software interrupts. FIRQV is used for the "fast" interrupt while IRQV and NMIV are used for maskable and non-maskable interrupts respectively. A special vector, RESTRT, is provided for re-entry into PSYMON. This permits the normally unmodifiable RESET vector to be redirected. Initially SWI2V, SWI3V, IRQV, and NMIV are set to perform a register dump and return to the PSYMON command prompt. FIRQV initially points to an RTI (return from interrupt) instruction. SWIV points to PSYMON's breakpoint routine.

PSYMON's repetoire of commands is easily changed or enhanced. The pointer USRTBL in PSYMON's RAM contains the address of an alternate command table. It is initialized to zero, indicating no alternate table exists. This table, if used, must be constructed according to certain conventions. The first byte must be a 1, the length of a command in bytes. Each entry consists of a single ASCII character (the command) followed by the two-byte address of the routine which performs the command function. The end of the table is signified by a byte with bit 7 on (typically FF). Since the user table, if present, is always searched first, any or all of PSYMON's commands may be redefined by the user.

Command routines should preserve the U and S registers and should exit via an RTS (return from subroutine). Approximately 38 bytes of stack are available via the S register. If a larger stack is required, the user routine must provide for it.

PSYMON I/O

As previously mentioned, all I/O within PSYMON is handled using a Device Control Block (DCB). To perform I/O using a DCB it is first necessary to construct the DCB. The minimum DCB is 10 bytes long containing the fields DCBLNK through DCBEXT. Other fields may be added (DCBAPP) as required by the device driver. Complete definitions of the DCB fields are contained in the PSYMON Advanced Programmer's Guide.

A caller wishing to perform I/O on a specific device must perform the following steps:

1. Load the A register with any driver parameter needed. (for example, the character to be outputted)
2. Load the B register with the I/O function code. (the I/O function code is described later)
3. Load the X register with the desired DCB address.
4. Call REQIO (JSR REQIO).

The driver routine may use B, X, and Y freely without saving them, as they are saved and restored by REQIO. Register A is used for passing results and parameters. Its contents, therefore, has meaning only to the driver and the caller.

Interpretation of the various I/O function codes is also up to the device driver. The codes currently defined are as follows:

Hex code	Meaning to driver
01	Read a physical record from device
02	Write a physical record to device
04	Return device status in A register
08	Perform control function to device

Functions 01 and 02 are straightforward, being simply the traditional read and write functions. The only real difference is what constitutes a physical record. In ACIA communication with a console a physical record is a single character. I/O with a disk may define a sector as the physical record.

Function 04 returns an 8-bit status in A with the following meanings:

Bit	Meaning if bit set to 1
0	Device has input ready.
1	Device can accept output.
2	Undefined.
3	Undefined.
4	Undefined.
5	Undefined.
6	Undefined.
7	Device is inoperative or in standby.

The use of this function is dependent on the device. In an ACIA driver it might be used to test for a 'break' request, while in a disk driver it could be used to detect a write-protect condition.

The final function defined, 08, is used to perform certain non-data related control functions on a device. In the ACIA driver within PSYMON this function is used to perform the configuration functions necessary for an ACIA. Here again the function's meaning is dependent on the driver's interpretation of it.

PSYMON SUBROUTINES

One of the design goals of PSYMON was to provide a good monitor with a rich supply of useful subroutines which could be easily used by programmers writing "system" programs. A concerted effort was made to construct useful tools that could be built upon rather than requiring the re-invention of similar functions. The subroutines discussed in this section have all been designed to be called externally. Any subroutine not mentioned here was designed for a specific purpose within PSYMON and should not be considered as a general-purpose routine. The subroutines are discussed in the order of their occurrence within PSYMON.

SEARCH - General table search.

This routine is designed to search a table of words and addresses. The word length must be fixed and is given in the first byte of the table. Addresses are two bytes long. The last byte of the table should be FF (hex). On entry register Y must point to the first byte of the item to be located in the table. Register X must point at the first byte of the table to be searched. Upon exit from this routine the Z flag, if set, indicates a successful outcome and X points to the address corresponding to the word which matched. If the Z flag is clear the item could not be located and register X points to the end sentinel of the table. Registers A and B are altered by this routine.

COMPAR - General string compare.

This routine compares two strings of arbitrary but equal length. The condition code flags are set as a result of the compare. On entry X contains the address of string 1, Y contains the address of string 2, and B contains the string length. On exit B, X, and Y are unchanged while A is altered.

LOAD - Load a hex program.

This program is designed to load a program in S1-S9 format. Input characters are obtained using the DCB pointed to by CIDCB. If CEDCB is non-zero the incoming characters will be echoed to the device whose DCB it points to. All registers are modified except U and S. The outcome of the load is reflected in the CKSUM variable in PSYMON RAM. If CKSUM is zero it indicates a successful load with an S9 termination. A non-zero value means an illegal character was encountered, a RAM error occured, or a checksum was invalid.

GETHEX - Get hexadecimal number from console.

This routine gets characters from the console (using CIDCB) to build a hexadecimal number in X. On exit A contains the last character entered (terminator), B contains a count of hex characters processed, and X contains the hex number right justified with zero fill. The Z flag is set if no hex digits were encountered, clear otherwise. Other registers are preserved.

INHEX - Input hex digit from console.

This routine inputs a character from the console (using CIDCB) and checks it for a legal hexadecimal digit. If legal the digit is converted into binary. If not the character is unchanged. The Z flag is set if the character is non-hex, clear otherwise. Registers X, Y, U, and S are unchanged.

INCHR - Input character from console.

A character is read from the console (using CIDCB) and returned in the A register. Except for C no other registers are changed. The character is stripped of parity and echoed if necessary (using CEDCB, if non-zero).

OUTCHR - Output character to console.

The character in A is output to the console (using CODCB). Only the C register is changed.

REQIO - Perform I/O request.

On entry X must point to the DCB for the device to be accessed. Register B contains the function code to be performed, while A contains a driver parameter, if required. On exit the A register may contain a driver result, depending on the function. All other registers are preserved except C.

DSPDBY - Display double byte and space.

The content of registers A and B is displayed on the console (using CODCB) as hex digits (A most significant byte) followed by a space. All registers are preserved except C.

DSPSBY - Display single byte and space.

The content of the A register is displayed on the console (using CODCB) as two hex digits followed by a space. Only the C register is altered.

OUTSP - Output a space to the console.

A single space is output to the console (using CODCB). No registers are altered except C.

OUTHEX - Output A register as 2 hex digits.

The contents of the A register are displayed on the console (using CODCB) as two hex digits. Only the C register is altered.

PSTRNG - Display string on console.

On entry X points to the string to be displayed. Characters are displayed successively (using CODCB) until a character is encountered which has bit 7 turned on. This character is also displayed (with bit 7 masked off) and the routine exits with X pointing to the next character past the end of the string. Registers A, X, and C are changed.

CRLF - Do carriage return/line feed on console.

A carriage return and line feed are output to the console (using CODCB). Only C is altered. Note that no nulls are output following this sequence. If a device requires nulls following this sequence the device driver must provide them.

SAVE - Save a program in S1 format.

The beginning and ending addresses to be saved must be in BEGADD and ENDADD prior to calling SAVE. Output is done using CODCB. No S9 is output. This should be done by the caller if it is required. All registers are changed except U and S.

FURTHER INFORMATION

Further information regarding PSYMON may be obtained by examination of the PSYMON assembly listing. Users requiring unique modifications to PSYMON may submit their requirements to Percom Data Company for a quotation.

```
       NAM   PSYMON

*************************************************
* PSYMON VERSION 1.20                           *
* A 6809 ROM MONITOR                            *
*                                               *
* THE PERCOM SYSTEM MONITOR (PSYMON) WAS        *
* WRITTEN BY A TEAM OF PROGRAMMERS USING        *
* STRUCTURED TECHNIQUES.  THE TEAM MEMBERS      *
* ARE AS FOLLOWS:                               *
*     HAROLD A MAUCH - PRESIDENT, PERCOM DATA   *
*     MIKE FOREMAN - 6809 PROJECT LEADER        *
*     BYRON SEASTRUNK - DESIGN ENGINEER         *
*     CLIFF RUSHING - PROGRAMMER                *
*     JIM STUTSMAN - CHIEF PROGRAMMER           *
*                                               *
* COPYRIGHT (c) 1979 PERCOM DATA COMPANY, INC.  *
* USE OF THIS SOFTWARE IS GRANTED ROYALTY-FREE  *
* AS LONG AS THE USER CLEARLY ACKNOWLEDGES ITS  *
* ORIGIN.                                       *
*                                               *
* WHILE THIS MONITOR IS VERY SIMPLE, ITS TRUE   *
* POWER LIES IN ITS EXTENSIBILITY AND IN THE    *
* TOOLS THAT IT PROVIDES FOR OTHER SOFTWARE     *
* TO USE.  THIS OPERATING SYSTEM IS DEDICATED   *
* TO HAROLD MAUCH AND HIS LEGENDARY 512 BYTE    *
* OPERATING SYSTEM.                             *
*                                               *
* COMMANDS:                                     *
*   M <ADDRESS> - MEMORY EXAMINE/CHANGE         *
*   G <ADDRESS> - GO TO ADDRESS                 *
*   R <REGISTER> - REGISTER EXAMINE/CHANGE      *
*   L - LOAD PROGRAM FROM TAPE                  *
*   S <START> <END> - SAVE PROGRAM TO TAPE      *
*   B <ADDRESS> - SET/LIST BREAKPOINTS          *
*   U <ADDRESS> - UNSET BREAKPOINTS             *
*   Z - JUMP TO PROM AT ADDRESS C000 HEX        *
*                                               *
* CALLABLE SUBROUTINES:                         *
*   INCHR - INPUT CHARACTER FROM CONSOLE        *
*   OUTCHR - OUTPUT CHARACTER TO CONSOLE        *
*   REQIO - PERFORM I/O TO PERIPHERAL           *
*   GETHEX - INPUT HEX NUMBER FROM CONSOLE      *
*   INHEX - INPUT HEX DIGIT FROM CONSOLE        *
*   DSPSBY - DISPLAY SINGLE BYTE & SPACE        *
*   DSPDBY - DISPLAY DOUBLE BYTE & SPACE        *
*   OUTHEX - DISPLAY 2 HEX DIGIST               *
*   PSTRNG - DISPLAY STRING ON CONSOLE          *
*   LOAD - LOAD HEX PROGRAM FROM CONSOLE        *
*   SAVE - SAVE HEX PROGRAM TO CONSOLE          *
*   CRLF - BEGIN NEW LINE ON CONSOLE            *
*   OUTS - OUTPUT SPACE TO CONSOLE              *
*                                               *
* ALL I/O WITHIN PSYMON IS DONE THROUGH THE     *
```

```
                * USE OF DEVICE CONTROL BLOCKS.  THIS ALLOWS    *
                * EASY MODIFICATION BY THE USER.  PSYMON HAS    *
                * FOUR DCB POINTERS INITIALIZED TO POINT TO THE *
                * CONSOLE (ACIA) DCB.  THEY ARE USED AS         *
                * FOLLOWS:                                      *
                *   CIDCB - POINTS TO DCB USED FOR CONSOLE      *
                *           INPUT (CHARACTER I/O).              *
                *   CEDCB - POINTS TO DCB USED FOR ECHO OF      *
                *           CHARACTERS RECEIVED USING CIDCB.    *
                *           ECHO MAY BE SUPPRESSED BY SETTING   *
                *           THIS POINTER TO ZERO.               *
                *   CODCB - POINTS TO DCB USED FOR CONSOLE      *
                *           OUTPUT (CHARACTER I/O).             *
                *   TPDCB - POINTS TO DCB USED FOR PSYMON       *
                *           TAPE LOAD & SAVE COMMANDS.          *
                *                                               *
                * THE PSYMON COMMAND TABLE MAY BE EXTENDED      *
                * OR CHANGED BY SETTING THE POINTER 'USRTBL'    *
                * TO THE ADDRESS OF A USER COMMAND TABLE.  IT   *
                * IS INITIALIZED TO ZERO, INDICATING NO USER    *
                * TABLE EXISTS.                                 *
                *                                               *
                * ADDITIONAL INFORMATION REGARDING THE USE OF   *
                * 'PSYMON' MAY BE OBTAINED FROM:                *
                *     PERCOM DATA COMPANY, INC.                 *
                *     211 NORTH KIRBY                           *
                *     GARLAND, TEXAS  75042                     *
                *                                               *
                * REVISION A - 11/23/79                         *
                *   ADDITION OF A VECTOR FOR SCRATCHPAD RAM     *
                *                                               *
                * REVISION B - 02/08/80                         *
                *   ADDITION OF A VECTOR FOR FREE RAM           *
                *                                               *
                *************************************************

                * SYSTEM ADDRESS CONSTANTS
    FC00        ROM1   EQU   $FC00      BASE ADDRESS OF PSYMON ROM
    F800        ROM2   EQU   $F800      BASE ADDRESS OF EXTENSION ROM
    F380        RAM    EQU   $F380      BASE ADDRESS OF SCRATCHPAD RAM
    F000        FREE   EQU   $F000      ADDRESS OF FREE RAM
    F7FE        TERMNL EQU   $F7FE      SYSTEM TERMINAL ACIA


                * ASCII CHARACTER CONSTANTS
    000D        CR     EQU   $0D        CARRIAGE RETURN
    000A        LF     EQU   $0A        LINE FEED
    0020        SP     EQU   $20        SPACE

                * ACIA CONTROL CONFIGURATIONS
    0003        RESET  EQU   $03        RESET ACIA
    0051        CONFIG EQU   $51        SET FOR 8 DATA, 2 STOP, NO PARITY
    0011        RDRON  EQU   CONFIG-$40 READER ON (RTS ON)
    0051        RDROFF EQU   CONFIG     READER OFF (RTS OFF)
```

```
                      * PSYMON DCB OFFSETS
       0000           DCBLNK EQU   0          POINTER TO NEXT DCB IN CHAIN
       0002           DCBDID EQU   2          ASCII 2 CHARACTER DEVICE ID
       0004           DCBDVR EQU   4          DEVICE DRIVER ADDRESS
       0006           DCBIOA EQU   6          DEVICE I/O ADDRESS
       0008           DCBERR EQU   8          ERROR STATUS CODE
       0009           DCBEXT EQU   9          NUMBER OF EXTENSION BYTES IN DCB
       000A           DCBAPP EQU   10         DCB APPENDAGE FOR DRIVER USE

                      * PSYMON DCB FUNCTION CODES
       0001           READFN EQU   $01        READ FUNCTION CODE
       0002           WRITFN EQU   $02        WRITE FUNCTION CODE
       0004           STATFN EQU   $04        STATUS FUNCTION CODE
       0008           CNTLFN EQU   $08        DEVICE CONTROL FUNCTION CODE
```

```
00125                       * PSYMON RAM DEFINITIONS
00126 F380                         ORG   RAM

00128                       * PSYMON INTERNAL STACK & REGISTER SPACE
00129                       *   OFFSETS TO RAM BASE IN PARENTHESES
00130 F380      0037               RMB   55        STACK SPACE
00131           F3B7        STACK  EQU   *         (55) TOP OF STACK
00132 F3B7      0001        REGC   RMB   1         (55) CONDITION CODE REGISTER
00133 F3B8      0001        REGA   RMB   1         (56) A REGISTER
00134 F3B9      0001        REGB   RMB   1         (57) B REGISTER
00135 F3BA      0001        REGD   RMB   1         (58) DIRECT PAGE REGISTER
00136 F3BB      0002        REGX   RMB   2         (59) X REGISTER
00137 F3BD      0002        REGY   RMB   2         (61) Y REGISTER
00138 F3BF      0002        REGU   RMB   2         (63) U STACK POINTER
00139 F3C1      0002        REGP   RMB   2         (65) PROGRAM COUNTER

00141                       * PSYMON BREAKPOINT TABLE
00142 F3C3      000F        BPTABL RMB   15        (67) SPACE FOR 5 BREAKPOINTS
00143           F3D2        BPTEND EQU   *         (82) END OF BREAKPOINT TABLE

00145                       * PSYMON WORK AREAS
00146 F3D2      0002        MEMPTR RMB   2         (82) MEMORY POINTER FOR 'M' COMMAND
00147 F3D4      0002        USRTBL RMB   2         (84) ADDRESS OF USER COMMAND TABLE
00148 F3D6      0001        COMAND RMB   1         (86) COMMAND CHARACTER STORAGE
00149 F3D7      0001        CKSUM  RMB   1         (87) CHECKSUM FOR LOAD AND SAVE
00150 F3D8      0002        BEGADD RMB   2         (88) BEGIN ADDRESS FOR SAVE
00151 F3DA      0002        ENDADD RMB   2         (90) END ADDRESS FOR SAVE
00152 F3DC      0002        STKPTR RMB   2         (92) CONTENTS OF STACK POINTER

00154                       * THE PSYMON CONSOLE DCB
00155 F3DE      000A        CONDCB RMB   10        (94) STANDARD DCB

00157                       * PSYMON DCB POINTERS
00158 F3E8      0002        DCBCHN RMB   2         (104) BASE OF DCB CHAIN
00159 F3EA      0002        CIDCB  RMB   2         (106) CONSOLE INPUT DCB
00160 F3EC      0002        CEDCB  RMB   2         (108) CONSOLE ECHO DCB
00161 F3EE      0002        CODCB  RMB   2         (110) CONSOLE OUTPUT DCB
00162 F3F0      0002        TPDCB  RMB   2         (112) CASSETTE TAPE DCB

00164                       * PSYMON VECTORS
00165 F3F2      0002        SWI3V  RMB   2         (114) SOFTWARE INTERRUPT 3
00166 F3F4      0002        SWI2V  RMB   2         (116) SOFTWARE INTERRUPT 2
00167 F3F6      0002        FIRQV  RMB   2         (118) FAST INTERRUPT REQUEST
00168 F3F8      0002        IRQV   RMB   2         (120) INTERRUPT REQUEST
00169 F3FA      0002        SWIV   RMB   2         (122) SOFTWARE INTERRUPT
00170 F3FC      0002        NMIV   RMB   2         (124) NON-MASKABLE INTERRUPT
00171 F3FE      0002        FRERAM RMB   2         (126) ADDRESS OF FREE RAM
```

```
00173                           * PSYMON ROM CODING
00174 FC00                             ORG   ROM1
00175                           ************************************************
00176                           * PSYMON INITIALIZATION                        *
00177                           ************************************************
00178 FC00 10CE F3B7          4 INIT   LDS   #STACK    SET UP STACK POINTER
00179 FC04 1F   41            6        TFR   S,X       POINT X AT STACK
00180 FC06 6F   80            8 INIT1  CLR   ,X+       CLEAR A BYTE
00181 FC08 8C   F3E0          4        CMPX  #CONDCB+2  ALL FIELDS CLEAR?
00182 FC0B 26   F9            3        BNE   INIT1     LOOP IF NOT
00183 FC0D 108E FFBA          4        LDY   #RAMINT   POINT TO RAM DATA
00184 FC11 EC   A1            8 INIT2  LDD   ,Y++      MOVE 2 BYTES
00185 FC13 ED   81            8        STD   ,X++
00186 FC15 8C   F400          4        CMPX  #FRERAM+2  END OF RAM?
00187 FC18 26   F7            3        BNE   INIT2     LOOP IF NOT
00188 FC1A 8E   F3DE          3        LDX   #CONDCB   POINT TO DCB
00189 FC1D CC   0308          3        LDD   #RESET*256+CNTLFN  A=RESET, B=CNTLFN
00190 FC20 BD   FD63          8        JSR   REQIO     RESET ACIA
00191 FC23 86   51            2        LDA   #CONFIG   CONFIGURE ACIA
00192 FC25 BD   FD63          8        JSR   REQIO
00193 FC28 B6   F800          5        LDA   ROM2      CHECK FOR SECOND ROM
00194 FC2B 81   7E            2        CMPA  #$7E      IS THERE A JUMP THERE?
00195 FC2D 26   03            3        BNE   MONENT    GO IF NOT
00196 FC2F BD   F800          8        JSR   ROM2      CALL SECOND ROM

00198                           ************************************************
00199                           * PSYMON USER ENTRY                            *
00200                           ************************************************
00201 FC32 10FF F3DC          7 MONENT STS   STKPTR    SAVE STACK POINTER

00203                           ************************************************
00204                           * GET COMMAND                                  *
00205                           ************************************************
00206 FC36 8E   FC4A          3 GETCMD LDX   #PROMPT   DISPLAY PROMPT
00207 FC39 BD   FD97          8        JSR   PSTRNG
00208 FC3C BD   FD44          8        JSR   INCHR     INPUT COMMAND CHARACTER
00209 FC3F 8D   0F            7        BSR   LOOKUP    LOOK IT UP
00210 FC41 26   F3            3        BNE   GETCMD    LOOP IF NOT FOUND
00211 FC43 BD   FD75          8        JSR   OUTSP     OUTPUT A SPACE
00212 FC46 AD   94           10        JSR   [,X]      CALL COMMAND ROUTINE
00213 FC48 20   EC            3        BRA   GETCMD    GO BACK FOR MORE

00215 FC4A      0D              PROMPT FCB   CR,LF
      FC4B      0A
00216 FC4C      43                     FCC   'CMD'
      FC4D      4D
      FC4E      44
00217 FC4F      BF                     FCB   '?+$80    END OF STRING

00219                           ************************************************
00220                           * LOOK UP COMMAND IN TABLE                     *
00221                           ************************************************
00222 FC50 108E F3D6          4 LOOKUP LDY   #COMAND   POINT Y TO COMMAND
00223 FC54 A7   A4            4        STA   ,Y        SAVE COMMAND CHARACTER
```

```
00224 FC56 BE    F3D4     6        LDX   USRTBL   GET USER TABLE ADDRESS
00225 FC59 27    04       3        BEQ   LOOK1    GO IF NONE
00226 FC5B 8D    05       7        BSR   SEARCH   SEARCH USER TABLE
00227 FC5D 27    10       3        BEQ   SERCHX   GO IF FOUND
00228 FC5F 8E    FFA3     3 LOOK1  LDX   #CMDTBL  SEARCH INTERNAL TABLE

00230                       **************************************************
00231                       * GENERAL TABLE SEARCH                           *
00232                       *                                                *
00233                       * ENTRY REQUIREMENTS:  X - POINTS TO TABLE       *
00234                       *                      Y - POINTS TO ITEM        *
00235                       *                      FIRST BYTE OF TABLE MUST  *
00236                       *                      CONTAIN ITEM LENGTH       *
00237                       *                      LAST BYTE MUST BE FF      *
00238                       *                                                *
00239                       * EXIT CONDITIONS:  C - Z SET IF FOUND, CLEAR    *
00240                       *                       IF NOT FOUND             *
00241                       *                   X - POINTS TO ADDRESS OF     *
00242                       *                       ROUTINE FOR MATCH        *
00243                       *                   A,B - CHANGED                *
00244                       *                                                *
00245                       **************************************************
00246 FC62 E6    80       6 SEARCH LDB   ,X+      GET ITEM LENGTH
00247 FC64 8D    0A       7 SERCH1 BSR   COMPAR   COMPARE CURRENT ITEM
00248 FC66 3A             3        ABX            ADVANCE TO NEXT ITEM
00249 FC67 27    06       3        BEQ   SERCHX   EXIT IF MATCH
00250 FC69 30    02       5        LEAX  2,X      STEP OVER ADDRESS
00251 FC6B 6D    84       6        TST   ,X       END OF TABLE?
00252 FC6D 2A    F5       3        BPL   SERCH1   LOOP IF NOT
00253 FC6F 39             5 SERCHX RTS

00255                       **************************************************
00256                       * GENERAL STRING COMPARE                         *
00257                       *                                                *
00258                       * ENTRY REQUIREMENTS:  X - ADDRESS OF STRING 1   *
00259                       *                      Y - ADDRESS OF STRING 2   *
00260                       *                      B - LENGTH OF STRINGS     *
00261                       *                                                *
00262                       * EXIT CONDITIONS:  C - SET PER COMPARE 1:2      *
00263                       *                   B,X,Y - UNCHANGED            *
00264                       *                   A - CHANGED                  *
00265                       *                                                *
00266                       **************************************************
00267 FC70 34    34       9 COMPAR PSHS  B,X,Y    SAVE REGISTERS
00268 FC72 A6    80       6 COMP1  LDA   ,X+      GET NEXT CHARACTER
00269 FC74 A1    A0       6        CMPA  ,Y+      COMPARE IT
00270 FC76 26    03       3        BNE   COMP2    EXIT IF UNMATCHED
00271 FC78 5A             2        DECB           DECREMENT LOOP COUNT
00272 FC79 26    F7       3        BNE   COMP1
00273 FC7B 35    B4      11 COMP2  PULS  B,X,Y,PC RESTORE REGISTERS & EXIT

00275                       **************************************************
00276                       * LOAD PROGRAM FROM TAPE                         *
00277                       **************************************************
```

```
00278 FC7D FC    F3EA      6 TLOAD  LDD   CIDCB     SAVE CONSOLE DCBS
00279 FC80 BE    F3EC      6        LDX   CEDCB
00280 FC83 34    16        8        PSHS  A,B,X
00281 FC85 BE    F3F0      6        LDX   TPDCB     POINT TO TAPE DCB
00282 FC88 4F              2        CLRA            SET D TO 0
00283 FC89 5F              2        CLRB
00284 FC8A BF    F3EA      6        STX   CIDCB     SET TAPE IN, NO ECHO
00285 FC8D FD    F3EC      6        STD   CEDCB
00286 FC90 CC    1108      3        LDD   #RDRON*256+CNTLFN  RAISE READER CONTROL
00287 FC93 BD    FD63      8        JSR   REQIO
00288 FC96 8D    1B        7        BSR   LOAD      LOAD THE TAPE
00289 FC98 CC    5108      3        LDD   #RDROFF*256+CNTLFN  DROP READ CONTROL
00290 FC9B BE    F3F0      6        LDX   TPDCB
00291 FC9E BD    FD63      8        JSR   REQIO
00292 FCA1 35    16        8        PULS  A,B,X     RESTORE CONSOLE DCBS
00293 FCA3 FD    F3EA      6        STD   CIDCB
00294 FCA6 BF    F3EC      6        STX   CEDCB
00295 FCA9 7D    F3D7      7        TST   CKSUM     ANY ERRORS?
00296 FCAC 27    45        3        BEQ   LOADX     GO IF NOT

00298                        ************************************************
00299                        * DISPLAY ERROR INDICATOR OF '?'               *
00300                        ************************************************
00301 FCAE 86    3F        2 ERROR  LDA   #'?       DISPLAY ERROR INDICATOR
00302 FCB0 7E    FD58      4        JMP   OUTCHR

00304                        ************************************************
00305                        * LOAD PROGRAM IN HEX FORMAT                   *
00306                        *                                              *
00307                        * ENTRY REQUIREMENTS:  NONE                    *
00308                        *                                              *
00309                        * EXIT CONDITIONS:  ALL REGISTERS CHANGED      *
00310                        *                   CKSUM NON-ZERO IF ERROR    *
00311                        *                                              *
00312                        ************************************************
00313 FCB3 1F    42        6 LOAD   TFR   S,Y       MARK STACK FOR ERROR RECOVERY
00314 FCB5 BD    FD44      8 LOAD1  JSR   INCHR     GET A CHARACTER
00315 FCB8 81    53        2 LOAD2  CMPA  #'S       START OF RECORD?
00316 FCBA 26    F9        3        BNE   LOAD1     LOOP IF NOT
00317 FCBC BD    FD44      8        JSR   INCHR     GET ANOTHER CHARACTER
00318 FCBF 81    39        2        CMPA  #'9       END OF LOAD?
00319 FCC1 27    30        3        BEQ   LOADX     GO IF YES
00320 FCC3 81    31        2        CMPA  #'1       START OF RECORD?
00321 FCC5 26    F3         BNE   LOAD2     LOOP IF NOT
00322 FCC7 7F    F3D7      7        CLR   CKSUM     INIT CHECKSUM
00323 FCCA 8D    28        7        BSR   INBYTE    READ LENGTH
00324 FCCC 80    02        2        SUBA  #2        ADJUST IT
00325 FCCE 1F    89        6        TFR   A,B       SAVE IN B
00326 FCD0 8D    22        7        BSR   INBYTE    GET ADDRESS HI
00327 FCD2 A7    E3        7        STA   ,--S      SAVE ON STACK
00328 FCD4 8D    1E        7        BSR   INBYTE    GET ADDRESS LO
00329 FCD6 A7    61        5        STA   1,S       PUT ON STACK
00330 FCD8 35    10        6        PULS  X         ADDRESS NOW IN X
00331 FCDA 8D    18        7 LOAD3  BSR   INBYTE    READ A BYTE
```

```
00332 FCDC 5A            2        DECB              DECREMENT COUNT
00333 FCDD 27   08       3        BEQ   LOAD4       GO IF DONE
00334 FCDF A7   84       4        STA   ,X          STORE BYTE
00335 FCE1 A1   80       6        CMPA  ,X+         VERIFY GOOD STORE
00336 FCE3 26   07       3        BNE   LOAD5       GO IF ERROR
00337 FCE5 20   F3       3        BRA   LOAD3
00338 FCE7 7C   F3D7     7 LOAD4  INC   CKSUM       CHECK CHECKSUM
00339 FCEA 27   C9       3        BEQ   LOAD1       LOOP IF GOOD
00340 FCEC 86   FF       2 LOAD5  LDA   #$FF        SET ERROR FLAG
00341 FCEE B7   F3D7     5        STA   CKSUM
00342 FCF1 1F   24       6        TFR   Y,S         RESTORE STACK
00343 FCF3 39            5 LOADX  RTS

00345                      ************************************************
00346                      * INPUT BYTE                                   *
00347                      ************************************************
00348 FCF4 8D   33       7 INBYTE BSR   INHEX       GET HEX DIGIT
00349 FCF6 27   EF       3        BEQ   LOAD4       GO IF ERROR
00350 FCF8 48            2        ASLA              SHIFT TO MS HALF
00351 FCF9 48            2        ASLA
00352 FCFA 48            2        ASLA
00353 FCFB 48            2        ASLA
00354 FCFC 34   02       5        PSHS  A           SAVE DIGIT
00355 FCFE 8D   29       7        BSR   INHEX       GET ANOTHER DIGIT
00356 FD00 27   E5       3        BEQ   LOAD4       GO IF ERROR
00357 FD02 AB   E4       4        ADDA  ,S          COMBINE HALVES
00358 FD04 A7   E4       4        STA   ,S          SAVE ON STACK
00359 FD06 BB   F3D7     5        ADDA  CKSUM       ADD TO CHECKSUM
00360 FD09 B7   F3D7     5        STA   CKSUM
00361 FD0C 35   82       7        PULS  A,PC        GET RESULT & RETURN

00363                      ************************************************
00364                      * GET HEX NUMBER FROM CONSOLE                  *
00365                      *                                              *
00366                      * ENTRY REQUIREMENTS:  NONE                    *
00367                      *                                              *
00368                      * EXIT CONDITIONS:  A - LAST CHAR INPUT        *
00369                      *                   B - HEX DIGIT COUNT        *
00370                      *                   X - HEX NUMBER             *
00371                      *                   C - SET ACCORDING TO B     *
00372                      *                                              *
00373                      ************************************************
00374 FD0E 5F            2 GETHEX CLRB              INITIALIZE DIGIT COUNT, RESULT
00375 FD0F 8E   0000     3        LDX   #0
00376 FD12 8D   15       7 GETHX1 BSR   INHEX       GET A DIGIT
00377 FD14 27   11       3        BEQ   GETHX2      GO IF NOT HEX
00378 FD16 1E   01       7        EXG   D,X         OLD RESULT TO A,B
00379 FD18 58            2        ASLB              SHIFT LEFT 1 DIGIT
00380 FD19 49            2        ROLA
00381 FD1A 58            2        ASLB
00382 FD1B 49            2        ROLA
00383 FD1C 58            2        ASLB
00384 FD1D 49            2        ROLA
00385 FD1E 58            2        ASLB
```

```
00386 FD1F 49             2        ROLA
00387 FD20 1E   01        7        EXG   D,X       REPLACE RESULT
00388 FD22 30   86        5        LEAX  A,X       ADD IN NEW DIGIT
00389 FD24 5C             2        INCB            ADD TO DIGIT COUNT
00390 FD25 20   EB        3        BRA   GETHX1    LOOP FOR MORE
00391 FD27 5D             2 GETHX2 TSTB            SET/RESET Z FLAG
00392 FD28 39             5        RTS

00394                       **************************************************
00395                       * GET HEX DIGIT FROM CONSOLE                     *
00396                       *                                                *
00397                       * ENTRY REQUIREMENTS:  NONE                      *
00398                       *                                                *
00399                       * EXIT CONDITIONS:  A - HEX DIGIT OR NON-HEX     *
00400                       *                   C - Z FLAG SET IF A NOT HEX *
00401                       *                   ALL OTHER REGS PRESERVED     *
00402                       *                                                *
00403                       **************************************************
00404 FD29 8D   19        7 INHEX  BSR   INCHR     GET A CHARACTER
00405 FD2B 34   02        5        PSHS  A         SAVE IT
00406 FD2D 80   30        2        SUBA  #$30      CONVERT TO BINARY
00407 FD2F 2B   0E        3        BMI   INHEX2    GO IF NOT NUMERIC
00408 FD31 81   09        2        CMPA  #$09      GREATER THAN 9?
00409 FD33 23   06        3        BLS   INHEX1    GO IF NOT
00410 FD35 80   07        2        SUBA  #$07      CONVERT LETTER
00411 FD37 81   0A        2        CMPA  #$0A      LEGAL VALUE?
00412 FD39 25   04        3        BLO   INHEX2    GO IF NOT
00413 FD3B 81   0F        2 INHEX1 CMPA  #$0F      GREATER THAN 15?
00414 FD3D 23   02        3        BLS   INHEX3    GO IF NOT
00415 FD3F A6   E4        4 INHEX2 LDA   ,S        GET ORIGINAL CHAR BACK
00416 FD41 A1   E0        6 INHEX3 CMPA  ,S+       SET/RESET Z FLAG
00417 FD43 39             5        RTS

00419                       **************************************************
00420                       * CONSOLE INPUT ROUTINE                          *
00421                       *                                                *
00422                       * ENTRY REQUIREMENTS:  NONE                      *
00423                       *                                                *
00424                       * EXIT CONDITIONS:  A - CHARACTER WITH PARITY    *
00425                       *                       REMOVED                  *
00426                       *                   ALL OTHER REGS PRESERVED     *
00427                       *                   EXCEPT C                     *
00428                       *                                                *
00429                       **************************************************
00430 FD44 34   14        7 INCHR  PSHS  B,X       SAVE REGISTERS
00431 FD46 BE   F3EA      6        LDX   CIDCB     POINT TO INPUT DCB
00432 FD49 C6   01        2        LDB   #READFN   SET UP FOR READ
00433 FD4B 8D   16        7        BSR   REQIO     READ A CHARACTER
00434 FD4D 84   7F        2        ANDA  #$7F      REMOVE PARITY
00435 FD4F BE   F3EC      6        LDX   CEDCB     POINT TO ECHO DCB
00436 FD52 34   02        5        PSHS  A         SAVE CHARACTER
00437 FD54 26   07        3        BNE   OUTCH1    GO IF ECHO
00438 FD56 35   96       10        PULS  A,B,X,PC  RESTORE & RETURN
```

```
                     *************************************************
                     * CONSOLE OUTPUT ROUTINE                        *
                     *                                               *
                     * ENTRY REQUIREMENTS:  A - CHARACTER TO BE      *
                     *                          OUTPUT TO CONSOLE    *
                     *                                               *
                     * EXIT CONDITIONS:  ALL REGISTERS PRESERVED     *
                     *                   EXCEPT C                    *
                     *                                               *
                     *************************************************
FD58 34   16        8 OUTCHR PSHS  A,B,X    SAVE REGISTERS
FD5A BE   F3EE      6        LDX   CODCB    POINT TO OUTPUT DCB
FD5D C6   02        2 OUTCH1 LDB   #WRITFN  SET FUNCTION
FD5F 8D   02        7        BSR   REQIO    OUTPUT THE CHARACTER
FD61 35   96       10        PULS  A,B,X,PC RESTORE REGISTERS & RETURN

                     *************************************************
                     * PERFORM I/O REQUESTS                          *
                     *                                               *
                     * ENTRY REQUIREMENTS:  A - DRIVER PARAMETER     *
                     *                      B - FUNCTION CODE        *
                     *                      X - DCB ADDRESS          *
                     *                                               *
                     * EXIT CONDITIONS:  A - DRIVER RESULT           *
                     *                   ALL OTHERS PRESERVED        *
                     *                   EXCEPT C                    *
                     *                                               *
                     *************************************************
FD63 34   7C       12 REQIO  PSHS  B,DP,X,Y,U  SAVE REGISTERS
FD65 AD   98 04    12        JSR   [DCBDVR,X]  CALL DRIVER
FD68 35   FC       14        PULS  B,DP,X,Y,U,PC  RESTORE REGISTERS & EXIT

                     *************************************************
                     * DISPLAY DOUBLE BYTE                           *
                     *                                               *
                     * ENTRY REQUIREMENTS:  A,B - DOUBLE BYTE        *
                     *                            TO BE PRINTED      *
                     *                                               *
                     * EXIT CONDITIONS:  ALL REGISTERS PRESERVED     *
                     *                   EXCEPT C                    *
                     *                                               *
                     *************************************************
FD6A 8D   11        7 DSPDBY BSR   OUTHEX   DISPLAY A AS 2 HEX DIGITS
FD6C 1E   89        7        EXG   A,B      LS BYTE TO A
FD6E 8D   03        7        BSR   DSPSBY   DISPLAY AS 2 DIGITS, SPACE
FD70 1E   89        7        EXG   A,B      RESTORE A & B
FD72 39             5        RTS

                     *************************************************
                     * DISPLAY A BYTE AND SPACE                      *
                     *                                               *
                     * ENTRY REQUIREMENTS:  A - BYTE TO BE DISPLAYED *
                     *                                               *
                     * EXIT CONDITIONS:  ALL REGISTERS PRESERVED     *
```

```
00494                       *                   EXCEPT C                     *
00495                       *                                                *
00496                       **************************************************
00497 FD73 8D   08        7 DSPSBY BSR   OUTHEX   DISPLAY BYTE IN A

00499                       **************************************************
00500                       * OUTPUT A SPACE TO THE CONSOLE                  *
00501                       *                                                *
00502                       * ENTRY REQUIREMENTS:  NONE                      *
00503                       *                                                *
00504                       * EXIT CONDITIONS:  ALL REGISTERS PRESERVED      *
00505                       *                   EXCEPT C                     *
00506                       *                                                *
00507                       **************************************************
00508 FD75 34   02        5 OUTSP  PSHS  A        SAVE A REGISTER
00509 FD77 86   20        2        LDA   #SP      OUTPUT A SPACE

00511                       **************************************************
00512                       * OUTPUT CHARACTER, RESTORE A, & RETURN          *
00513                       **************************************************
00514 FD79 8D   DD        7 OUTCHX BSR   OUTCHR   DISPLAY CHARACTER
00515 FD7B 35   82        7        PULS  A,PC     RESTORE & EXIT

00517                       **************************************************
00518                       * DISPLAY A REGISTER AS 2 HEX DIGITS             *
00519                       *                                                *
00520                       * ENTRY REQUIREMENTS:  A - BYTE TO DISPLAY       *
00521                       *                                                *
00522                       * EXIT CONDITIONS:  ALL REGISTERS PRESERVED      *
00523                       *                   EXCEPT C                     *
00524                       *                                                *
00525                       **************************************************
00526 FD7D 34   02        5 OUTHEX PSHS  A        SAVE THE BYTE
00527 FD7F 44             2        LSRA           GET MS DIGIT
00528 FD80 44             2        LSRA
00529 FD81 44             2        LSRA
00530 FD82 44             2        LSRA
00531 FD83 8D   06        7        BSR   OUTDIG   DISPLAY IT
00532 FD85 A6   E4        4        LDA   ,S       GET LS DIGIT
00533 FD87 8D   02        7        BSR   OUTDIG   DISPLAY IT
00534 FD89 35   82        7        PULS  A,PC     RESTORE A & RETURN

00536                       **************************************************
00537                       * DISPLAY A HEX DIGIT                            *
00538                       **************************************************
00539 FD8B 84   0F        2 OUTDIG ANDA  #$0F     MASK OFF DIGIT
00540 FD8D 8B   30        2        ADDA  #$30     CONVERT TO ASCII
00541 FD8F 81   39        2        CMPA  #$39     BIGGER THAN 9?
00542 FD91 23   C5        3        BLS   OUTCHR   GO IF NOT
00543 FD93 8B   07        2        ADDA  #$07     CONVERT TO LETTER
00544 FD95 20   C1        3        BRA   OUTCHR   PRINT AND EXIT

00546                       **************************************************
00547                       * PRINT A STRING TO THE CONSOLE                  *
```

```
                          *                                                *
                          * ENTRY CONDITIONS:  X - POINTS TO STRING        *
                          *                    LAST BYTE HAS BIT 7 ON      *
                          *                                                *
                          * EXIT CONDITIONS:  X - POINTS 1 BYTE PAST END   *
                          *                   A,C - CHANGED                *
                          *                                                *
                          **************************************************
FD97 A6    84           4 PSTRNG LDA    ,X        GET A CHARACTER
FD99 84    7F           2        ANDA   #$7F      MASK OFF
FD9B 8D    BB           7        BSR    OUTCHR    DISPLAY IT
FD9D 6D    80           8        TST    ,X+       WAS IT LAST?
FD9F 2A    F6           3        BPL    PSTRNG    LOOP IF NOT
FDA1 39                 5        RTS

                          **************************************************
                          * PRINT CR/LF ON CONSOLE                         *
                          *                                                *
                          * ENTRY REQUIREMENTS:  NONE                      *
                          *                                                *
                          * EXIT CONDITIONS:  ALL REGISTERS PRESERVED      *
                          *                   EXCEPT C                     *
                          *                                                *
                          **************************************************
FDA2 34    02           5 CRLF   PSHS   A         SAVE A REGISTER
FDA4 86    0D           2        LDA    #CR       OUTPUT CR
FDA6 8D    B0           7        BSR    OUTCHR
FDA8 86    0A           2        LDA    #LF       OUTPUT LF & EXIT
FDAA 20    CD           3        BRA    OUTCHX

                          **************************************************
                          * SAVE PROGRAM ON TAPE                           *
                          **************************************************
FDAC 8D    30           7 TSAVE  BSR    GETHX     GET START ADDRESS
FDAE 27    0E           3        BEQ    TSAVE2    GO IF NONE
FDB0 BF    F3D8         6        STX    BEGADD    SAVE START
FDB3 8D    29           7        BSR    GETHX     GET END ADDRESS
FDB5 26    04           3        BNE    TSAVE1    GO IF ENTERED
FDB7 BE    F3D8         6        LDX    BEGADD    DUPLICATE ADDRESS
FDBA 5C                 2        INCB             SET ADDRESS INDICATOR
FDBB BF    F3DA         6 TSAVE1 STX    ENDADD    SAVE END
FDBE BE    F3EE         6 TSAVE2 LDX    CODCB     SAVE CONSOLE DCB
FDC1 34    12           7        PSHS   A,X       SAVE TERMINATOR TOO
FDC3 BE    F3F0         6        LDX    TPDCB     SET UP FOR TAPE
FDC6 BF    F3EE         6        STX    CODCB
FDC9 5D                 2        TSTB             ANY ADDRESS ENTERED?
FDCA 27    02           3        BEQ    TSAVE3    GO IF NOT
FDCC 8D    13           7        BSR    SAVE      SAVE THE PROGRAM
FDCE 35    02           5 TSAVE3 PULS   A         GET TERMINATOR
FDD0 81    0D           2        CMPA   #CR       WAS IT RETURN?
FDD2 26    04           3        BNE    TSAVE4    GO IF NOT
FDD4 C6    39           2        LDB    #'9       OUTPUT S9 RECORD
FDD6 8D    54           7        BSR    OUTSN
FDD8 35    10           6 TSAVE4 PULS   X         RESTORE DCB POINTER
```

```
00602 FDDA BF   F3EE      6         STX   CODCB
00603 FDDD 39             5         RTS

00605                       **************************************************
00606                       * GET HEX NUMBER IN X                            *
00607                       **************************************************
00608 FDDE 7E   FD0E      4 GETHX   JMP   GETHEX    RELATIVE BRANCH BOOSTER

00610                       **************************************************
00611                       * SAVE A PROGRAM IN HEX                          *
00612                       *                                                *
00613                       * ENTRY REQUIREMENTS:  SAVE ADDRESSES ARE IN     *
00614                       *                      BEGADDR & ENDADDR         *
00615                       *                                                *
00616                       * EXIT CONDITIONS:  ALL REGISTERS CHANGED        *
00617                       *                                                *
00618                       **************************************************
00619 FDE1 BE   F3D8      6 SAVE    LDX   BEGADD    POINT AT FIRST BYTE
00620 FDE4 C6   31        2 SAVE1   LDB   #'1       BEGIN NEW S1 RECORD
00621 FDE6 8D   44        7         BSR   OUTSN
00622 FDE8 7F   F3D7      7         CLR   CKSUM     INIT CHECKSUM
00623 FDEB FC   F3DA      6         LDD   ENDADD    CALCULATE BYTES TO SAVE
00624 FDEE 34   10        6         PSHS  X
00625 FDF0 A3   E1        9         SUBD  ,S++
00626 FDF2 4D             2         TSTA            GREATER THAN 255?
00627 FDF3 26   04        3         BNE   SAVE2     GO IF YES
00628 FDF5 C1   10        2         CMPB  #16       LESS THAN FULL RECORD?
00629 FDF7 25   02        3         BLO   SAVE3     GO IF YES
00630 FDF9 C6   0F        2 SAVE2   LDB   #15       SET FULL RECORD SIZE
00631 FDFB 5C             2 SAVE3   INCB            CORRECT RECORD SIZE
00632 FDFC 1F   98        6         TFR   B,A       OUTPUT RECORD SIZE
00633 FDFE 8B   03        2         ADDA  #3        ADJUST FOR ADDRESS,COUNT
00634 FE00 8D   20        7         BSR   OUTBYT
00635 FE02 34   10        6         PSHS  X         ADDRESS TO STACK
00636 FE04 35   02        5         PULS  A         OUTPUT ADDRESS HI
00637 FE06 8D   1A        7         BSR   OUTBYT
00638 FE08 35   02        5         PULS  A         OUTPUT ADDRESS LO
00639 FE0A 8D   16        7         BSR   OUTBYT
00640 FE0C A6   80        6 SAVE4   LDA   ,X+       SAVE A DATA BYTE
00641 FE0E 8D   12        7         BSR   OUTBYT
00642 FE10 5A             2         DECB            LOOP UNTIL 0
00643 FE11 26   F9        3         BNE   SAVE4
00644 FE13 B6   F3D7      5         LDA   CKSUM     GET CHECKSUM
00645 FE16 43             2         COMA            COMPLIMENT IT
00646 FE17 8D   09        7         BSR   OUTBYT    OUTPUT IT
00647 FE19 31   1F        5         LEAY  -1,X      CHECK FOR END
00648 FE1B 10BC F3DA      8         CMPY  ENDADD
00649 FE1F 26   C3        3         BNE   SAVE1     LOOP IF NOT
00650 FE21 39             5         RTS

00652                       **************************************************
00653                       * OUTPUT BYTE AS HEX AND ADD TO CHECKSUM         *
00654                       **************************************************
00655 FE22 BD   FD7D      8 OUTBYT  JSR   OUTHEX    OUTPUT BYTE AS HEX
```

```
00656 FE25 BB   F3D7      5        ADDA  CKSUM    ADD TO CHECKSUM
00657 FE28 B7   F3D7      5        STA   CKSUM
00658 FE2B 39             5        RTS

00660                       ***********************************************
00661                       * OUTPUT 'S' TAPE RECORD HEADERS              *
00662                       ***********************************************
00663 FE2C BD   FDA2      8 OUTSN  JSR   CRLF     BEGIN NEW LINE
00664 FE2F 86   53        2        LDA   #'S      OUTPUT 'S' HEADER
00665 FE31 8D   02        7        BSR   OUTC
00666 FE33 1F   98        6        TFR   B,A      RECORD TYPE TO A

00668                       ***********************************************
00669                       * OUTPUT CHARACTER TO CONSOLE                 *
00670                       ***********************************************
00671 FE35 7E   FD58      4 OUTC   JMP   OUTCHR   RELATIVE BRANCH BOOSTER

00673                       ***********************************************
00674                       * MEMORY EXAMINE AND CHANGE                   *
00675                       ***********************************************
00676 FE38 8D   A4        7 MEMEC  BSR   GETHX    GET ADDRESS
00677 FE3A 26   03        3        BNE   MEMEC1   GO IF GOOD
00678 FE3C BE   F3D2      6        LDX   MEMPTR   USE PREVIOUS
00679 FE3F BF   F3D2      6 MEMEC1 STX   MEMPTR   UPDATE RAM POINTER
00680 FE42 BD   FDA2      8        JSR   CRLF     BEGIN NEW LINE
00681 FE45 1F   10        6        TFR   X,D      DISPLAY ADDRESS
00682 FE47 BD   FD6A      8        JSR   DSPDBY
00683 FE4A A6   80        6        LDA   ,X+      GET CONTENTS
00684 FE4C BD   FD73      8        JSR   DSPSBY   DISPLAY THEM
00685 FE4F 1F   12        6        TFR   X,Y      SAVE ADDRESS IN Y
00686 FE51 8D   8B        7        BSR   GETHX    GET CHANGE DATA
00687 FE53 1E   01        7        EXG   D,X      SAVE DELIM, GET NEW
00688 FE55 27   09        3        BEQ   MEMEC2   GO IF NO CHANGE
00689 FE57 E7   3F        5        STB   -1,Y     UPDATE MEMORY
00690 FE59 E1   3F        5        CMPB  -1,Y     VERIFY GOOD STORE
00691 FE5B 27   03        3        BEQ   MEMEC2   GO IF GOOD STORE
00692 FE5D BD   FCAE      8        JSR   ERROR    DISPLAY ERROR
00693 FE60 1F   10        6 MEMEC2 TFR   X,D      GET DELIMITER IN A
00694 FE62 1F   21        6        TFR   Y,X      GET NEXT ADDRESS IN X
00695 FE64 81   0D        2        CMPA  #CR      END OF UPDATE?
00696 FE66 27   08        3        BEQ   MEMEC3   GO IF YES
00697 FE68 81   5E        2        CMPA  #'^      BACKING UP?
00698 FE6A 26   D3        3        BNE   MEMEC1   LOOP IF NOT
00699 FE6C 30   83        7        LEAX  ,--X     BACK UP 2
00700 FE6E 20   CF        3        BRA   MEMEC1   CONTINUE
00701 FE70 39             5 MEMEC3 RTS

00703                       ***********************************************
00704                       * GO TO ADDRESS                               *
00705                       ***********************************************
00706 FE71 10FE F3DC      7 GO     LDS   STKPTR   SET UP STACK
00707 FE75 BD   FD0E      8        JSR   GETHEX   GET TARGET ADDRESS
00708 FE78 27   02        3        BEQ   GO1      GO IF NONE
00709 FE7A AF   6A        6        STX   10,S     STORE IN PC ON STACK
```

```
00710 FE7C A6   E4        4 GO1    LDA   ,S        SET 'E' FLAG IN CC
00711 FE7E 8A   80        2        ORA   #$80
00712 FE80 A7   E4        4        STA   ,S
00713 FE82 3B            15 INTRET RTI             LOAD REGISTERS AND GO

00715                       ***********************************************
00716                       * BREAKPOINT (SOFTWARE INTERRUPT) TRAP        *
00717                       ***********************************************
00718 FE83 AE   6A        6 BRKPNT LDX   10,S      GET PROGRAM COUNTER
00719 FE85 30   1F        5        LEAX  -1,X      DECREMENT BY 1
00720 FE87 AF   6A        6        STX   10,S      REPLACE ON STACK
00721 FE89 C6   FF        2        LDB   #$FF      FLAG FOR SINGLE REMOVAL
00722 FE8B BD   FF43      8        JSR   REMBK     REMOVE BREAKPOINT

00724                       ***********************************************
00725                       * INTERRUPT (HARDWARE/SOFTWARE) TRAP          *
00726                       ***********************************************
00727 FE8E 10FF F3DC      7 TRAP   STS   STKPTR    SAVE STACK POINTER
00728 FE92 BD   FDA2      8        JSR   CRLF      BEGIN NEW LINE
00729 FE95 8D   3C        7        BSR   REGDMP    DUMP REGISTERS
00730 FE97 7E   FC36      4        JMP   GETCMD    GET NEXT COMMAND

00732                       ***********************************************
00733                       * REGISTER EXAMINE AND CHANGE                 *
00734                       ***********************************************
00735 FE9A BD   FD44      8 REGEC  JSR   INCHR     GET REGISTER TO EXAMINE
00736 FE9D BD   FDA2      8        JSR   CRLF      BEGIN NEW LINE
00737 FEA0 5F             2        CLRB            CLEAR OFFSET COUNT
00738 FEA1 8E   FEC7      3        LDX   #REGIDS   POINT TO REGISTER ID STRING
00739 FEA4 A1   85        5 REGEC1 CMPA  B,X       CHECK REGISTER NAME
00740 FEA6 27   07        3        BEQ   REGEC2    GO IF FOUND
00741 FEA8 5C             2        INCB            ADVANCE COUNTER
00742 FEA9 C1   0B        2        CMPB  #11       END OF LIST?
00743 FEAB 23   F7        3        BLS   REGEC1    LOOP IF NOT
00744 FEAD 20   24        3        BRA   REGDMP    BAD ID - DUMP ALL
00745 FEAF 34   04        5 REGEC2 PSHS  B         SAVE OFFSET
00746 FEB1 8D   37        7        BSR   RDUMP     DISPLAY THE REG & CONTENTS
00747 FEB3 BD   FD0E      8        JSR   GETHEX    GET NEW VALUE
00748 FEB6 35   04        5        PULS  B         RESTORE OFFSET
00749 FEB8 27   0C        3        BEQ   REGECX    GO IF NO CHANGE
00750 FEBA 31   A5        5        LEAY  B,Y       POINT TO REG ON STACK
00751 FEBC C1   03        2        CMPB  #3        SINGLE BYTE REG?
00752 FEBE 1F   10        6        TFR   X,D       GET NEW DATA IN A,B
00753 FEC0 23   02        3        BLS   REGEC3    GO IF SINGLE
00754 FEC2 A7   A0        6        STA   ,Y+       STORE MS BYTE
00755 FEC4 E7   A4        4 REGEC3 STB   ,Y        STORE LS BYTE
00756 FEC6 39             5 REGECX RTS

00758 FEC7      43          REGIDS FCC   'CABDXXYYUUPP'
      FEC8      41
      FEC9      42
      FECA      44
      FECB      58
      FECC      58
      FECD      59
      FECE      59
      FECF      55
      FED0      55
      FED1      50
      FED2      50

00760                           ****************************************************
00761                           * COMPLETE REGISTER DUMP                           *
00762                           ****************************************************
00763 FED3 8E   FEC7      3 REGDMP LDX   #REGIDS   POINT TO ID STRING
00764 FED6 5F             2        CLRB            CLEAR OFFSET COUNTER
00765 FED7 A6   85        5 RGDMP1 LDA   B,X       GET REG NAME
00766 FED9 8D   0F        7        BSR   RDUMP     DISPLAY IT
00767 FEDB 5C             2        INCB            BUMP TO NEXT REG
00768 FEDC C1   0B        2        CMPB  #11       ALL PRINTED?
00769 FEDE 23   F7        3        BLS   RGDMP1    LOOP IF NOT
00770 FEE0 86   53        2        LDA   #'S       DISPLAY STACK ID
00771 FEE2 8D   1B        7        BSR   DSPID
00772 FEE4 108E F3D0      4        LDY   #STKPTR-12  Y+B=>STKPTR
00773 FEE8 20   0A        3        BRA   RDUMP1

00775                           ****************************************************
00776                           * DISPLAY REGISTER CONTENTS                        *
00777                           ****************************************************
00778 FEEA 8D   13        7 RDUMP  BSR   DSPID     DISPLAY REGISTER ID
00779 FEEC 10BE F3DC      7        LDY   STKPTR    POINT Y AT STACK
00780 FEF0 C1   03        2        CMPB  #3        SINGLE BYTE REG?
00781 FEF2 23   06        3        BLS   RDUMP2    GO IF YES
00782 FEF4 A6   A5        5 RDUMP1 LDA   B,Y       DISPLAY MS BYTE
00783 FEF6 BD   FD7D      8        JSR   OUTHEX
00784 FEF9 5C             2        INCB            ADVANCE OFFSET
00785 FEFA A6   A5        5 RDUMP2 LDA   B,Y       DISPLAY A BYTE
00786 FEFC 7E   FD73      4        JMP   DSPSBY

00788                           ****************************************************
00789                           * DISPLAY REGISTER ID                              *
00790                           ****************************************************
00791 FEFF 8D   02        7 DSPID  BSR   OUTCH     DISPLAY REG NAME
00792 FF01 86   3D        2        LDA   #'=       DISPLAY '='

00794                           ****************************************************
00795                           * OUTPUT CHARACTER TO CONSOLE                      *
00796                           ****************************************************
00797 FF03 7E   FD58      4 OUTCH  JMP   OUTCHR    RELATIVE BRANCH BOOSTER

00799                           ****************************************************
00800                           * SET A BREAKPOINT                                 *
00801                           ****************************************************
00802 FF06 BD   FD0E      8 SETBK  JSR   GETHEX    GET ADDRESS
00803 FF09 27   18        3        BEQ   DSPBK     GO IF NONE ENTERED
00804 FF0B 8D   27        7        BSR   INITBP    POINT Y AT BP TABLE
00805 FF0D EC   A4        5 SETBK1 LDD   ,Y        EMPTY SLOT?
00806 FF0F 27   06        3        BEQ   SETBK2    GO IF YES
```

```
00807 FF11 8D   26       7        BSR    NEXTBP   ADVANCE TO NEXT SLOT
00808 FF13 26   F8       3        BNE    SETBK1   LOOP IF NOT END
00809 FF15 20   0C       3        BRA    DSPBK    EXIT
00810 FF17 AF   A4       5 SETBK2 STX    ,Y       SAVE ADDRESS
00811 FF19 27   08       3        BEQ    DSPBK    GO IF ADDRESS = 0
00812 FF1B A6   84       4        LDA    ,X       GET CONTENTS
00813 FF1D A7   22       5        STA    2,Y      SAVE IN TABLE
00814 FF1F 86   3F       2        LDA    #$3F     SWI OP CODE
00815 FF21 A7   84       4        STA    ,X       SET BREAK

00817                      *************************************************
00818                      * DISPLAY ALL BREAKPOINTS                       *
00819                      *************************************************
00820 FF23 BD   FDA2     8 DSPBK  JSR    CRLF     BEGIN NEW LINE
00821 FF26 8D   0C       7        BSR    INITBP   POINT Y AT BP TABLE
00822 FF28 EC   A4       5 DSPBK1 LDD    ,Y       GET ADDRESS OF BP
00823 FF2A 27   03       3        BEQ    DSPBK2   GO IF INACTIVE
00824 FF2C BD   FD6A     8        JSR    DSPDBY   DISPLAY ADDRESS
00825 FF2F 8D   08       7 DSPBK2 BSR    NEXTBP   ADVANCE POINTER
00826 FF31 26   F5       3        BNE    DSPBK1   LOOP IF NOT END
00827 FF33 39            5        RTS

00829                      *************************************************
00830                      * INITIALIZE BREAKPOINT TABLE POINTER           *
00831                      *************************************************
00832 FF34 108E F3C3     4 INITBP LDY    #BPTABL  POINT Y AT BP TABLE
00833 FF38 39            5        RTS

00835                      *************************************************
00836                      * ADVANCE BREAKPOINT TABLE POINTER              *
00837                      *************************************************
00838 FF39 31   23       5 NEXTBP LEAY   3,Y      ADVANCE TO NEXT ENTRY
00839 FF3B 108C F3D2     5        CMPY   #BPTEND  CHECK FOR END OF TABLE
00840 FF3F 39            5        RTS

00842                      *************************************************
00843                      * UNSET A BREAKPOINT                            *
00844                      *************************************************
00845 FF40 BD   FD0E     8 UNSBK  JSR    GETHEX   GET ADDRESS

00847                      *************************************************
00848                      * REMOVE ONE OR MORE BREAKPOINTS                *
00849                      *************************************************
00850 FF43 8D   EF       7 REMBK  BSR    INITBP   POINT Y AT BP TABLE
00851 FF45 5D            2 REMBK1 TSTB            REMOVE ALL?
00852 FF46 27   06       3        BEQ    REMBK2   GO IF YES
00853 FF48 AC   A4       6        CMPX   ,Y       FIND ADDRESS?
00854 FF4A 27   09       3        BEQ    UNSET    GO IF YES
00855 FF4C 20   02       3        BRA    REMBK3   LOOP IF NO
00856 FF4E 8D   05       7 REMBK2 BSR    UNSET    UNSET IT
00857 FF50 8D   E7       7 REMBK3 BSR    NEXTBP   ADVANCE POINTER
00858 FF52 26   F1       3        BNE    REMBK1   LOOP IF NOT END
00859 FF54 39            5        RTS
```

```
00861                          ************************************************
00862                          * REMOVE A BREAKPOINT                          *
00863                          ************************************************
00864 FF55 AE   A4        5 UNSET  LDX   ,Y        GET ADDRESS OF BP
00865 FF57 27   08        3        BEQ   UNSET1    GO IF INACTIVE
00866 FF59 A6   22        5        LDA   2,Y       GET CONTENTS
00867 FF5B A7   84        4        STA   ,X        REPLACE BP
00868 FF5D 6F   A4        6        CLR   0,Y       MARK BP INACTIVE
00869 FF5F 6F   21        7        CLR   1,Y
00870 FF61 39             5 UNSET1 RTS

00872                          ************************************************
00873                          * TERMINAL DRIVER (ACIA)                       *
00874                          ************************************************
00875 FF62 6F   08        7 TERMDR CLR   DCBERR,X NO ERRORS POSSIBLE
00876 FF64 AE   06        6        LDX   DCBIOA,X GET I/O ADDRESS
00877 FF66 54             2        LSRB            READ FUNCTION?
00878 FF67 25   0C        3        BCS   TERMRD    GO IF YES
00879 FF69 54             2        LSRB            WRITE FUNCTION?
00880 FF6A 25   11        3        BCS   TERMWT    GO IF YES
00881 FF6C 54             2        LSRB            STATUS FUNCTION?
00882 FF6D 25   17        3        BCS   TERMST    GO IF YES
00883 FF6F 54             2        LSRB            CONTROL FUNCTION?
00884 FF70 24   02        3        BCC   TERM1     GO IF NOT
00885 FF72 A7   84        4        STA   ,X        STORE CONTROL CODE
00886 FF74 39             5 TERM1  RTS

00888 FF75 E6   84        4 TERMRD LDB   ,X        GET STATUS
00889 FF77 54             2        LSRB            INPUT BIT TO C
00890 FF78 24   FB        3        BCC   TERMRD    LOOP IF NO INPUT
00891 FF7A A6   01        5        LDA   1,X       GET CHARACTER
00892 FF7C 39             5        RTS

00894 FF7D E6   84        4 TERMWT LDB   ,X        GET STATUS
00895 FF7F C5   02        2        BITB  #2        READY FOR OUTPUT?
00896 FF81 27   FA        3        BEQ   TERMWT    LOOP IF NOT
00897 FF83 A7   01        5        STA   1,X       OUTPUT CHARACTER
00898 FF85 39             5        RTS

00900 FF86 A6   84        4 TERMST LDA   ,X        GET STATUS
00901 FF88 84   03        2        ANDA  #3        MASK OFF READY BITS
00902 FF8A 39             5        RTS

00904                          ************************************************
00905                          * INTERRUPT HANDLERS                           *
00906                          ************************************************
00907 FF8B 6E   9F F3F2   9 SWI3   JMP   [SWI3V]   SOFTWARE INTERRUPT 3
00908 FF8F 6E   9F F3F4   9 SWI2   JMP   [SWI2V]   SOFTWARE INTERRUPT 2
00909 FF93 6E   9F F3F6   9 FIRQ   JMP   [FIRQV]   FAST INTERRUPT REQUEST
00910 FF97 6E   9F F3F8   9 IRQ    JMP   [IRQV]    INTERRUPT REQUEST
00911 FF9B 6E   9F F3FA   9 SWI    JMP   [SWIV]    SOFTWARE INTERRUPT
00912 FF9F 6E   9F F3FC   9 NMI    JMP   [NMIV]    NON-MASKABLE INTERRUPT

00914                          ************************************************
```

```
00915                      * PSYMON COMMAND TABLE                          *
00916                      *************************************************
00917 FFA3      01         CMDTBL FCB   1        ITEM LENGTH
00918 FFA4      4D                FCB   'M       MEMORY EXAMINE/CHANGE
00919 FFA5      FE38              FDB   MEMEC
00920 FFA7      47                FCB   'G       GOTO ADDRESS
00921 FFA8      FE71              FDB   GO
00922 FFAA      4C                FCB   'L       PROGRAM LOAD
00923 FFAB      FC7D              FDB   TLOAD
00924 FFAD      53                FCB   'S       PROGRAM SAVE
00925 FFAE      FDAC              FDB   TSAVE
00926 FFB0      52                FCB   'R       REGISTER EXAMINE/CHANGE
00927 FFB1      FE9A              FDB   REGEC
00928 FFB3      42                FCB   'B       SET/PRINT BREAKPOINTS
00929 FFB4      FF06              FDB   SETBK
00930 FFB6      55                FCB   'U       UNSET BREAKPOINTS
00931 FFB7      FF40              FDB   UNSBK
00932 FFB9      FF                FCB   $FF      END SENTINEL

00934                      *************************************************
00935                      * RAM INITIALIZATION DATA                       *
00936                      *************************************************
00937 FFBA      43         RAMINT FCC   'CN' CONSOLE DCB ID
      FFBB      4E
00938 FFBC      FF62              FDB   TERMDR   CONSOLE DRIVER
00939 FFBE      F7FE              FDB   TERMNL   CONSOLE I/O ADDRESS
00940 FFC0      0000              FDB   0        ERROR STATUS, EXT
00941 FFC2      F3DE              FDB   CONDCB   DCB CHAIN POINTER
00942 FFC4      F3DE              FDB   CONDCB   DCB POINTERS
00943 FFC6      F3DE              FDB   CONDCB
00944 FFC8      F3DE              FDB   CONDCB
00945 FFCA      F3DE              FDB   CONDCB
00946 FFCC      FE8E              FDB   TRAP     INTERRUPT VECTORS
00947 FFCE      FE8E              FDB   TRAP
00948 FFD0      FE82              FDB   INTRET
00949 FFD2      FE8E              FDB   TRAP
00950 FFD4      FE83              FDB   BRKPNT
00951 FFD6      FE8E              FDB   TRAP
00952 FFD8      F000              FDB   FREE

00954 FFDA      FF                FCB   $FF,$FF,$FF,$FF  RESERVED SPACE
      FFDB      FF
      FFDC      FF
      FFDD      FF

00956                      *************************************************
00957                      * SOFTWARE VECTORS                              *
00958                      *************************************************
00959 FFDE      F380              FDB   RAM      BASE OF PSYMON RAM
00960 FFE0      FD73              FDB   DSPSBY   DISPLAY SINGLE BYTE ON CONSOLE
00961 FFE2      FD6A              FDB   DSPDBY   DISPLAY DOUBLE BYTE ON CONSOLE
00962 FFE4      FD0E              FDB   GETHEX   GET HEX NUMBER FROM CONSOLE
00963 FFE6      FD97              FDB   PSTRNG   PRINT STRING TO CONSOLE
00964 FFE8      FD44              FDB   INCHR    INPUT CHARACTER FROM CONSOLE
```

```
00965 FFEA      FD58              FDB   OUTCHR   OUTPUT CHARACTER TO CONSOLE
00966 FFEC      FD63              FDB   REQIO    PERFORM I/O REQUEST
00967 FFEE      FC32              FDB   MONENT   MONITOR RE-ENTRY

00969                      **************************************************
00970                      * HARDWARE VECTORS                               *
00971                      **************************************************
00972 FFF0      FC00              FDB   INIT     RESERVED BY MOTOROLA
00973 FFF2      FF8B              FDB   SWI3     SOFTWARE INTERRUPT 3
00974 FFF4      FF8F              FDB   SWI2     SOFTWARE INTERRUPT 2
00975 FFF6      FF93              FDB   FIRQ     FAST INTERRUPT REQUEST
00976 FFF8      FF97              FDB   IRQ      INTERRUPT REQUEST
00977 FFFA      FF9B              FDB   SWI      SOFTWARE INTERRUPT
00978 FFFC      FF9F              FDB   NMI      NON-MASKABLE INTERRUPT
00979 FFFE      FC00              FDB   INIT     RESTART

00981           0000              END

TOTAL ERRORS   00000
TOTAL WARNINGS 00000
```

PERCOM DATA CO. INC.
211 North Kirby
Garland, TX 75042
(214) 272-3421

NOTICE

All COMPUTER PROGRAMS sold or distributed by PERCOM DATA CO. INC. are sold or distributed on an AS-IS basis WITHOUT WARRANTY.

PERCOM DATA CO. INC. shall have no LIABILITY or responsibility to customers, or any other person or entity with respect to any LIABILITY, LOSS, OR DAMAGE caused or alleged to be caused directly or indirectly by equipment or computer programs sold by PERCOM DATA CO. INC. including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such equipment or computer programs.

Good data processing procedure dictates that the user test the program, run and test sample sets of data, and run the system in parallel with the system previously in use for a period of time adequate to insure that results of operation of the computer or program are satisfactory.

This program is the sole property of the author or PERCOM DATA CO. INC. and has been registered with the United States Copyright Office. Lawful users of this program may use the program themselves, but may not make copies or translations of the program in any form other than as necessary to use the program. It is a violation of the Federal Copyright Laws, punishable by fines and/or imprisonment, for anyone to Copy or Translate this program for any other purpose, including for purposes of resale, license or lease to others.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

PRINTED IN USA